6

RULE BASED EXPERT SYSTEM APPROACH TOWARDS THE RESOLUTION OF THE TOWER OF HANOI

By

SAMUEL OBADAN
Benson Idahosa University,
Benin City.

And

DR KINGSLEY OBAHIAGBON
Benson Idahosa University,
Benin City.

Abstract

The tower of Hanoi has been a topic every computer scientist has had to grapple with. Its implementation cuts across a fair range of options. One of the most popular is the use of an effective but non-intuitive method: The Recursion. Although the Recursive method has gained popularity, this paper presents an intuitive approach. By keeping the towers constant and maintaining focus on the disks, a 'Rule Based Expert System' poses as an excellent alternative for the resolution of the tower of Hanoi. A usability test on novice users shows significant advantage of the expert system approach over the traditional recursive approach.

Keywords: Expert system, Recursive system, Tower of Hanoi

The Tower of Hanoi also referred to as the Tower of Brahma or Lucas' Tower, (Hofstadter, Douglas R 1985) can be defined as a mathematical game or puzzle. It consists of three rods, also referred to as towers and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on tower 1, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack from tower 1 to tower 3 while keeping to the following fundamental rules (fig 1):

- Only one disk may be moved at a time.
- Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.

- No disk may be placed on top of a smaller disk.

Expert systems represent a practical application of artificial intelligence (AI) research that has been going on for almost the entire history of general-purpose computing. Much has been learned about how to store knowledge and combine what is known to derive new results and solve problems. Expert systems based on these ideas can take many forms. The rule-based system used in the implementation of the tower of Hanoi provides a non recursive and intuitive approach to provide solution.

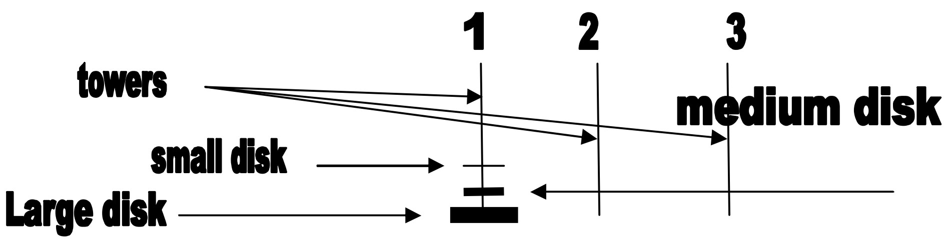

Fig. 1 Initial Setup of the Towers of Hanoi

The figure below (fig 2) demonstrates the sequence of iterations in moving 3 disks from tower 1 to tower 3 while keeping to the fundamental rules. At the end of the iterations, tower 3 would be stacked just as tower 1 was before the iterations began (at the start of the iterations). The iterations begin at ‘0’ and then terminate at ‘7’. The total number of moves must be equal to (2^n-1) where n represents the number of disks used for the tower of Hanoi.

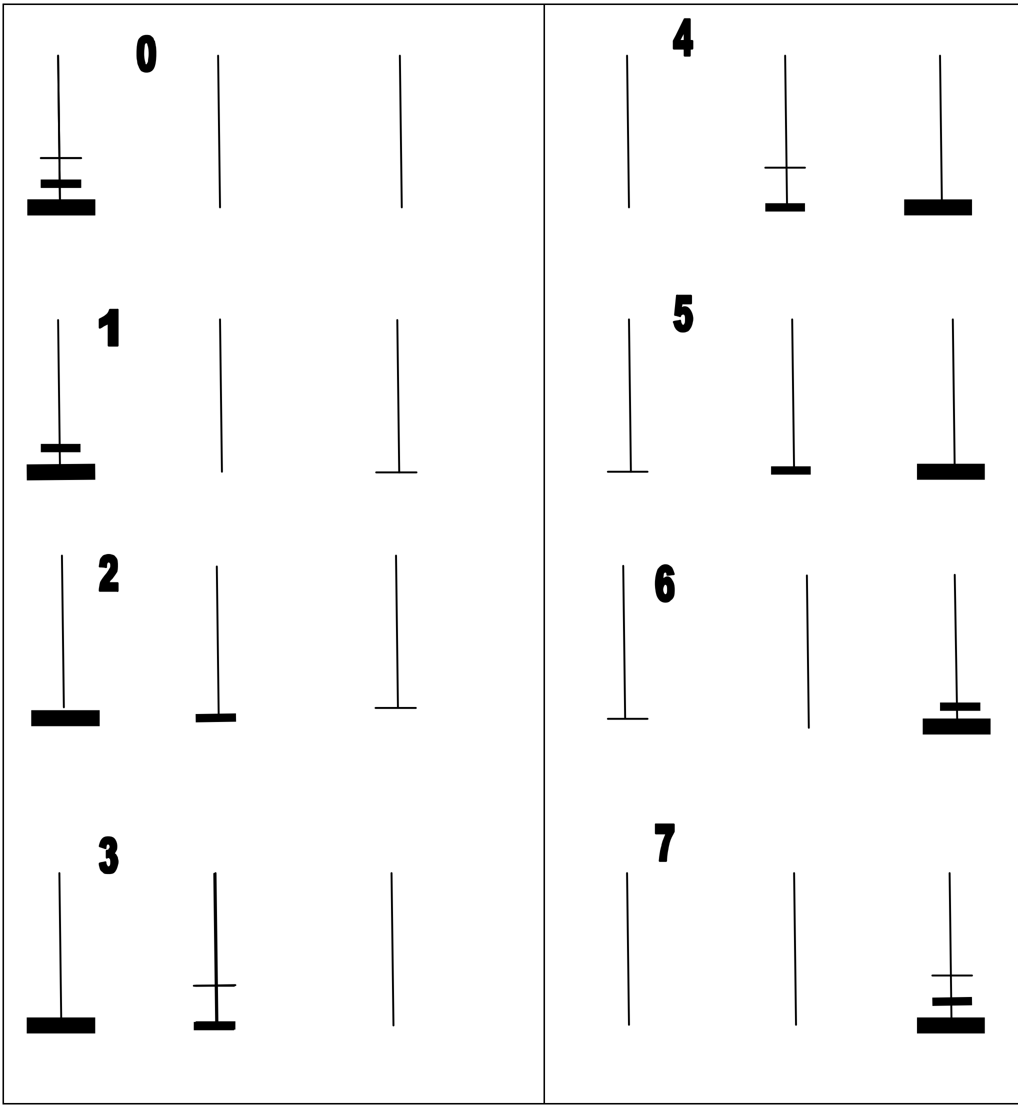

Fig 2 Transition Sequence of the Tower of Hanoi Using 3 Disks

Benefits of Expert Systems

- **Goal oriented:** Expert systems deliver answers to very specific questions that represent the *goals* of the system: they are not focused on abstract or theoretical information.

- **Efficient:** Requests for new information are based on the user's earlier responses –- no need wasting time providing irrelevant input.
- **Adaptive:** Even in situations where there are insufficient answers to all of the questions, alternate paths through the knowledge base might allow deduction of sufficient facts to provide useful advice.
- **Able to deal with uncertainty:** Expert systems are capable of processing uncertain responses and, by combining several pieces of uncertain information, may still be able to make strong recommendations.
- **Able to explain their information requests and suggestions:** Expert systems can also provide justification for each question asked along with a detailed explanation of the reasoning that led to any recommendations. (Michael Negnevitsky 2005)

Table 1 A Comparison of Rule Based Systems with Conventional Programs

RULE BASED SYSTEMS	CONVENTIONAL RECURSIVE PROGRAMS
Process knowledge expressed in the form of rules and use symbolic reasoning to solve Problems in a narrow domain.	Process data and use algorithms, a series of Well-defined operations, to solve general numerical problems.
Provide a clear separation of knowledge from its processing.	Do not separate knowledge from the control structure to process this knowledge.
Trace the rules fired during a problem-solving session and explain how a particular conclusion was reached and why specific data was needed.	Do not explain how a particular result was obtained and why input data was needed.
Permit inexact reasoning and can deal with Incomplete, uncertain and fuzzy data.	Work only on problems where data is complete and exact.
Enhance the quality of problem solving by adding new rules or adjusting old ones in the knowledge base. When new knowledge is acquired, changes are easy to accomplish.	Enhance the quality of problem solving by altering the program code. This affects both the knowledge and its processing, making changes difficult to implement.

Literature Review

There is a legend about an Indian temple which contains a large room with three time-worn posts in it surrounded by 64 golden disks. The Legend has it that Brahmin priests, acting out the command of an ancient prophecy, have been moving these disks, in accordance with the rules of the puzzle, since that time. The puzzle was therefore referred to as the Tower of Brahma puzzle. According to the legend, when the last move of the puzzle is completed, the world will end. (Ivan Moscovich 2001) However, there are many variations on this legend. For example, in some sightings, the temple is a monastery and the priests are monks. The temple or monastery may be said to be in different parts of the world — including Hanoi which is located at Vietnam. Also, in some versions, other elements are introduced, such as the fact that the tower was created at the beginning of the world, or that the priests or monks may make only one move per day.

A couple of solutions exist for the resolution of the Tower of Hanoi some of which include, the Iterative Solution, recursion, Binary solution, and Gray code solution.

Iterative Solution

The puzzle can be played with any number of disks, although many toy versions have around seven to nine of them. The game seems impossible to many novices, yet is solvable with a simple algorithm. The number of moves required to solve a Tower of Hanoi puzzle is 2^n -1, where *n* is the number of disks. (Petković, Miodrag 2009)

The following solution is a simple solution for the toy puzzle. Alternate moves between the smallest piece and a non-smallest piece. When moving the smallest piece, always move it to the next position in the same direction (to the right if the starting number of pieces is even, to the left if the starting number of pieces is odd). If there is no tower position in the chosen direction, move the piece to the opposite end, but then continue to move in the correct direction. For example, if you started with three pieces, you would move the smallest piece to the opposite end, then continue in the left direction after that. When the turn is to move the non-smallest piece, there is only one legal move. Doing this will complete the puzzle using the fewest number of moves to do so. It should perhaps be noted that this can be rewritten as a strikingly elegant set of rules.

Alternating between the smallest and the next-smallest disks, follow the steps for the appropriate case:

For an even number of disks:

- make the legal move between pegs A and B
- make the legal move between pegs A and C
- make the legal move between pegs B and C
- repeat until complete

For an odd number of disks:

- make the legal move between pegs A and C
- make the legal move between pegs A and B
- make the legal move between pegs B and C

- repeat until complete

In each case, a total of 2^n-1 moves are made.

Equivalent Iterative Solution

Another way to generate the unique optimal iterative solution is to number the disks 1 through n, largest to smallest. If n is odd, the first move is from the Start to the Finish peg, and if n is even the first move is from the Start to the Using peg.

Now, add the constraint that no odd disk may be placed directly on an odd disk, and no even disk may be placed directly on an even disk. With this extra constraint, and the obvious rule of never undoing your last move, there is only one move at every turn. The sequence of these unique moves is an optimal solution to the problem equivalent to the iterative solution described above. (Herbert Mayer, Don Perkins 1984)

Recursive Solution

A key to solving this puzzle is to recognize that it can be solved by breaking the problem down into a collection of smaller problems and further breaking those problems down into even smaller problems until a solution is reached. The following procedure demonstrates this approach.

- label the pegs A, B, C—these labels may move at different steps
- let *n* be the total number of discs
- number the discs from 1 (smallest, topmost) to *n* (largest, bottommost)

To move *n* discs from peg A to peg C:

1. move $n-1$ discs from A to B. This leaves disc *n* alone on peg A
2. move disc *n* from A to C
3. move $n-1$ discs from B to C so they sit on disc *n* (Dr. Kenneth A Lambert, Dr. Thomas Naps 2001)

The above is a recursive algorithm to carry out steps 1 and 3, apply the same algorithm again for $n-1$. The entire procedure is a finite number of steps, since at some point the algorithm will be required for $n = 1$. This step, moving a single disc from peg A to peg B, is trivial. This approach can be given a rigorous mathematical formalism with the theory of dynamic programming (Sniedovich, Moshe 2010) and is often used as an example of recursion when teaching programming.

Binary Solutions

Disk positions may be determined more directly from the binary (base 2) representation of the move number (the initial state being move #0, with all digits 0, and the final state being #2^n-1, with all digits 1), using the following rules:

1. There is one binary digit (bit) for each disk
2. The most significant (leftmost) bit represents the largest disk. A value of 0 indicates that the largest disk is on the initial peg, while a 1 indicates that it is on the final peg.
3. The bit string is read from left to right, and each bit can be used to determine the location of the corresponding disk.
4. A bit with the same value as the previous one means that the corresponding disk is stacked on top the previous disk on the same peg. (That is to say: a straight sequence of 1's or 0's means that the corresponding disks are all on the same peg).
5. A bit with a different value to the previous one means that the corresponding disk is one position to the left or right of the previous one. Whether it is left or right is determined by this rule:

- Assume that the initial peg is on the left and the final peg is on the right.
- Also assume "wrapping" - so the right peg counts as one peg "left" of the left peg, and vice versa.
- Let n be the number of greater disks that are located on the same peg as their first greater disk and add 1 if the largest disk is on the left peg. If n is even, the disk is located one peg to the left, if n is odd, the disk located one peg to the right.

For example, in an 8-disk Hanoi:

a. Move 0

- The largest disk is 0, so it is on the left (initial) peg.
- All other disks are 0 as well, so they are stacked on top of it. Hence all disks are on the initial peg.

b. Move 2^8-1

- The largest disk is 1, so it is on the right (final) peg.
- All other disks are 1 as well, so they are stacked on top of it. Hence all disks are on the final peg and the puzzle is complete.

c. Move 011011000 = 216_{10}

- The largest disk is 1, so it is on the right (final) peg.
- Disk two is also 1, so it is stacked on top of it, on the right peg.
- Disk three is 0, so it is on another peg. Since n is odd(n=3), it is one peg to the right, i.e. on the left peg.
- Disk four is 1, so it is on another peg. Since n is even(n=2), it is one peg to the left, i.e. on the right peg.
- Disk five is also 1, so it is stacked on top of it, on the right peg.
- Disk six is 0, so it is on another peg. Since n is odd(n=5), the disk is one peg to the right, i.e. on the left peg.
- Disk seven and eight are also 0, so they are stacked on top of it, on the left peg.

The source and destination pegs for the *m*th move can also be found elegantly from the binary representation of *m* using bitwise operations. To use the syntax of the C programming language, move *m* is from peg (m&m-1)%3 to peg ((m|m-1)+1)%3, where the disks begin on peg 0 and finish on peg 1 or 2 according to whether the number of disks is even or odd. Another formulation is from peg (m-(m&-m))%3 to peg (m+(m&-m))%3.

Furthermore the disk to be moved is determined by the number of times the move count (m) can be divided by 2 (i.e. the number of zero bits at the right), counting the first move as 1 and identifying the disks by the numbers 0, 1, 2 etc. in order of increasing size. This permits a very fast non-recursive computer implementation to find the positions of the disks after m moves without reference to any previous move or distribution of disks.

The count trailing zeros (ctz) operation, which counts the number of consecutive zeros at the end of a binary number, gives a simple solution to the problem: the disks are numbered from zero, and at move *m*, disk number ctz(*m*) is moved the minimum possible distance to the right (circling back around to the left as needed).

Gray Code Solution

The binary numeral system of Gray codes gives an alternative way of solving the puzzle. In the Gray system, numbers are expressed in a binary combination of 0s and 1s, but rather than being a standard positional numeral system, Gray code operates on the premise that each value differs from its predecessor by only one (and exactly one) bit changed. The number of bits present in Gray code is important, and leading zeros are not optional, unlike in positional systems.

If one counts in Gray code of a bit size equal to the number of disks in a particular Tower of Hanoi, begins at zero, and counts up, then the bit changed in each move corresponds to the disk to move, where the least-significant-bit is the smallest disk and the most-significant-bit is the largest.

Counting moves from 1 and identifying the disks by numbers starting from 0 in order of increasing size, the ordinal of the disk to be moved during move m is the number of times m can be divided by 2.

This technique identifies which disk to move, but not where to move it to. For the smallest disk there are always two possibilities. For the other disks there is always one possibility, except when all disks are on the same peg, but in that case either it is the smallest disk that must be moved or the objective has already been achieved. Luckily, there is a rule which does say where to move the smallest disk to. Let f be the starting peg, t the destination peg and r the remaining third peg. If the number of disks is odd, the smallest disk cycles along the pegs in the order f->t->r->f->t->r, etc. If the number of disks is even, this must be reversed: f->r->t->f->r->t etc.

Cyclic Hanoi

Cyclic Hanoi is a variation of the Hanoi in which each disk must be moved in the same cyclic direction, in most cases, clockwise. (T. D. Gedeon 1996) For example, given a standard three peg set-up, a given disk can be moved from peg A to peg B, then from B to C, C to A, etc. This can be solved using two mutually recursive procedures:

To move *n* discs **clockwise** from peg A to peg C:

1. move $n - 1$ discs **clockwise** from A to C
2. move disc #*n* from A to B
3. move $n - 1$ discs **counterclockwise** from C to A
4. move disc #*n* from B to C
5. move $n - 1$ discs **clockwise** from A to C

To move *n* discs **counterclockwise** from peg A to peg C:

1. move $n - 1$ discs **clockwise** from A to B
2. move disc #*n* from A to C
3. move $n - 1$ discs **clockwise** from B to C

Four Pegs and Beyond

Although the three-peg version has a simple recursive solution as outlined above, the *optimal* solution for the Tower of Hanoi problem with four pegs (called **Reve's puzzle**), let alone more pegs, is still an open problem. This is a good example of how a simple, solvable problem can be made dramatically more difficult by slightly loosening one of the problem constraints. The fact that the problem with four or more pegs is an open problem does not imply that no algorithm exists for finding (all of) the optimal solutions. Simply represent the game by an undirected graph, the nodes being distributions of disks and the edges being moves and use breadth first search to find one (or all) shortest path(s) moving a tower from one peg onto another one. However, even smartly implemented on the fastest computer now available, this algorithm provides no way of effectively computing solutions for large numbers of disks; the program would require more time and memory than available. Hence, even having an algorithm, it remains unknown how many moves an optimal solution requires and how many optimal solutions exist for 1000 disks and 10 pegs. Though it is not known exactly how many moves must be made, there are some asymptotic results. There is also a "presumed-optimal solution" given by the Frame-Stewart algorithm, discovered independently by Frame and Stewart in 1941. The related open Frame-Stewart conjecture claims that the Frame-Stewart algorithm always gives an optimal solution.

The optimality of the Frame-Stewart algorithm has been computationally verified for 4 pegs with up to 30 disks. (Korf, Richard E., Ariel Felner 2007)

An Expert System's Implementation

In the early 1970s, Newell and Simon from Carnegie-Mellon University proposed a production system model, the foundation of the modern rule-based expert systems. (Newell, A. and Simon, H.A 1972) The production model is based on the idea that humans solve problems by applying their knowledge (expressed as production rules) to a given problem represented by problem-specific information. The production rules are stored in the long-term memory and the problem-specific information or facts in the short-term memory. The production system model and the basic structure of a rule-based expert system are shown in figure 3. A rule-based expert system has five components: the knowledge base, the database, the inference engine, the explanation facilities, and the user interface.

The knowledge base contains the domain knowledge useful for problem solving. In a rule-based expert system, the knowledge is represented as a set of rules. Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the IF (condition) THEN (action) structure. When the condition part of a rule is satisfied, the rule is said to fire and the action part is executed. The database includes a set of facts used to match against the IF (condition) parts of rules stored in the knowledge base. The inference engine carries out the reasoning whereby the expert system reaches a solution. It links the rules given in the knowledge base with the facts provided in the database. The explanation facilities enable the user to ask the expert system how a particular conclusion is reached and why a specific fact is needed. An expert system must be able to explain its reasoning and justify its advice, analysis or conclusion. The user interface is the means of communication between a user seeking a solution to the problem and an expert system. The communication should be as meaningful and friendly as possible. These five components are essential for any rule-based expert system. They constitute its core, but there may be a few additional components.

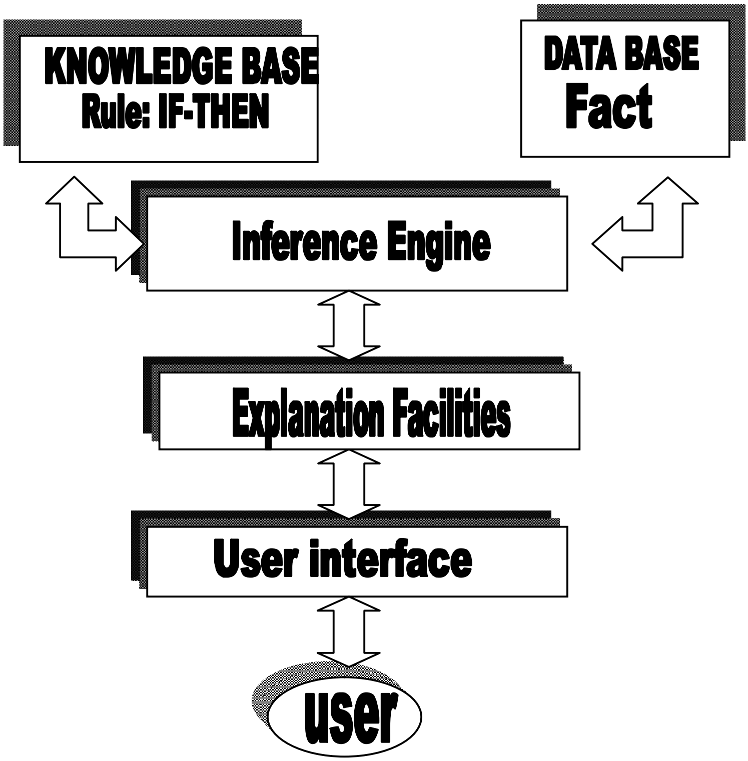

Figure 3 Production system and basic structure of a rule-based expert system: *Source* (Michael Negnevitsky 2005)

10 Commandments

1. A larger disk cannot come on top of a smaller one
2. The last recently moved disk cannot be moved in the immediate preceding sequence
3. The first move is instantiated from tower 1
4. If the number of disks in tower 1 is odd, and tower 1 is the source, then the disk destination tower is tower 3
5. If the number of disk in tower 1 is even, and tower 1 is the source, then the disk destination tower is tower 2
6. If the source is tower 2, and the number of disks on tower 2 is odd, then the disk destination tower is tower 3
7. If the source is tower 2, and the number of disks on tower 2 is even, then the disk destination tower is tower 1
8. If the source is tower 3, and the number of disks on tower 3 is odd, then the disk destination tower is tower 2

9. If the source is tower 3, and the number of disks on tower 3 is even, then the disk destination tower is tower 1
10. If the number of disks on tower 3 is equal to the total number of disks on tower 1 at instantiation, then the iteration ends.

At the end of this iterations keeping to the above 10 rules, the Tower 3 would be stacked just as Tower 1 was before the iteration began. The total number of moves must be (2^n-1) where n is the number of disks used for the tower of Hanoi.

Table 2 Decision Table for the Tower of Hanoi

RULES	1	2	3	4	5	6
Move from tower 1?	**Y**	**Y**	N	N	N	N
Move from tower 2?	N	N	**Y**	**Y**	N	N
Move from tower 3?	N	N	N	N	**Y**	**Y**
Odd no. of disks in tower 1	**Y**	N				
Even no. of disk in tower 1		**Y**				
Odd no. of disks in tower 2?			**Y**	N		
Even no. of disks in tower 2?			N	**Y**		
Odd no. of disks in tower 3?					**Y**	N
Even no. of disks in tower 3?					N	**Y**
No. of disks in tower 3 = initial no. of disks?	N	N	N	N	N	N
RESULTS						
Destination tower 1				**X**		**X**
Destination tower 2		**X**			**X**	
Destination tower 3	**X**		**X**			

Before the rules are fired, the inference engine ensures that, a larger disk cannot come on top of a smaller one (using weighted values as shown in fig 4 below) and also, the last recently moved disk cannot be moved in the immediate preceding sequence. An empty tower is considered to have a weight value of NULL. A tower with the NULL status can accept a disk of any weighted value.

After every move/iteration, the weight of each disk remains constant; however its position (location) on each tower after every move/iteration could either be ODD or EVEN. By keeping the towers constant, we concentrate on the movement of disks based on their eligibility. The following factors are responsible for a disk's eligibility for movement.

- ❖ It is the only available disk for movement

- It is not the last recently moved disk
- Its weight is less than the weights of the topmost disk in any or both of the other towers.

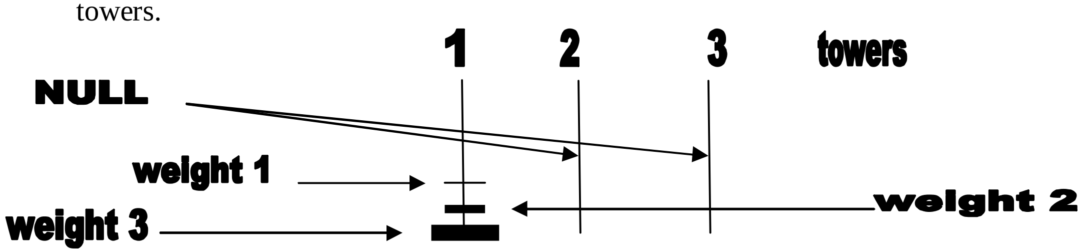

Fig 4 weighted value representation of the disks

Interestingly, there is never more than one eligible movable disk for each of the iteration; however there could be more than one possible destination tower for an eligible disk. The decision table adequately determines the destination tower which consequently enforces the maximum (2^n-1) iterations for n number of disks.

Algorithm for the Inference Engine

1. **INNITIALIZE** Counter = 0
2. **WHILE** (Initial no. of disks in Tower 1 != total no. of disks in Tower 3) **DO**
3. **IF** disk is eligible for movement and its position on the Tower is odd, **THEN** check decision table for destination Tower.
4. **IF** disk is eligible for movement and its position on the Tower is even, **THEN** check decision table for the destination Tower.
5. **INCREMENT** Counter
6. **LOOP**
7. **END**

A Recursive Implementation

A recursive approach towards the implementation of the tower of Hanoi is non-interactive and lacks explanation facilities which are the advantages of the expert system. Recursion implements the following algorithm: (See fig 4)

To move n disks from tower 1 to tower 3 using tower 2 as a temporary tower:

If (n > 0) Then

1. Move n-1 disks from tower 1 to tower 2 using tower 3 as a temporary tower
2. Move the remaining disk from tower 1 to tower 3
3. Move n-1 disks from tower 2 to tower 3 using tower 1 as a temporary tower

   End if

Methodology

In this paper, the usability of the expert system approach was tested and compared with the recursive system approach. The two systems were built and tested by a sample group of users and each user's feedback was captured with the aid of a questioner. Consequently, the level of usability of both systems was ascertained based on the technology acceptance model described below. (See Fig 5)

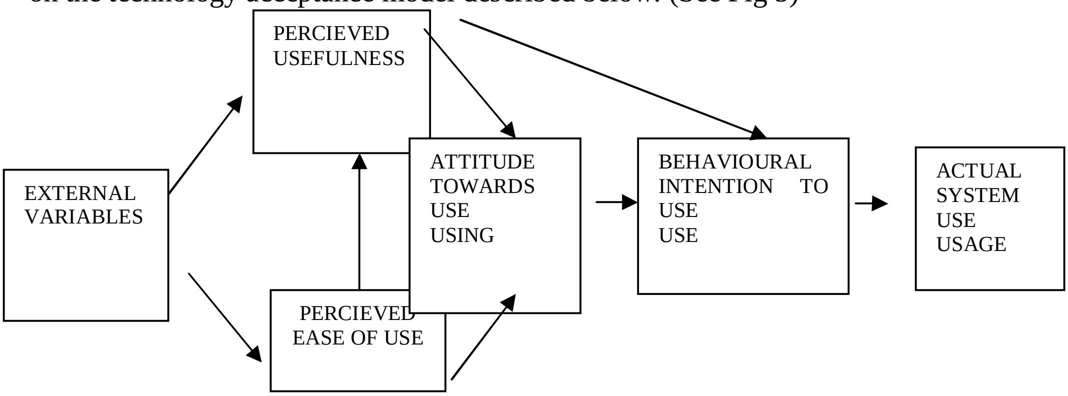

Fig 5 Technology Acceptance Model

(source FD Davis, RP Bagozzi, PR Warshaw 1989*)*

The usability of the two systems (expert system approach and recursive system approach) was evaluated and compared based on the above TAM (Technology Acceptance Model).

Data Collection

The data collection was conducted with the combination of a questioner and also by observation. The non-participant observer monitored the user activities without being involved in the participants' activities. Any vocal expression of frustration in using the system could be classified as negative while an expression of satisfaction could be classified as positive. The learn ability of the systems was focal in deciding the parameters for data collection. Broad view questioner which included all techniques of data collection in which each person is asked to respond to the same set of questions in a predetermined order was implemented for both systems.

Participant Sampling

Non-probability sampling methodology was employed for the selection of participants because they are available, convenient, and also represent some characteristic we wanted to study (novice users). Participants with good working knowledge of computers but ignorant to the concept of Hanoi were preferred. The participants were selected by the combination of both convenience (willingness and availability of participants) and snow sampling (participants invites other participants to become members of the sample) methodology.

Comparative Analysis

Central to the success of any system is adequate usability, given that the intended users have special needs. To this end, it was decided to use an empirical usability evaluation approach. Therefore, an experiment was conducted with a group of 100 novice participants of ages between 18 and 25 years.

Prior to the experiments, participants were given instructions and demonstration of their assigned systems through an example scenario. The participants were encouraged to ask questions at any time during the training scenario. The experiment involved each participant leveraging both systems to complete a prototype model of a 3 disk, 3 tower of Hanoi.

Upon training the group with the sample scenario for both systems, the evaluation scenario was invoked with each of the participants un-obtrusively allowed to undertake the task with the expert system and the recursive system in no particular order.

During the experiments, subjects' actions and mistakes were recorded along with their task completion times. At the end of the experiment, each participant was asked to complete a questioner that captured the users perceptions of both systems based on their experience. The following dependent behavioral variables were used for analysis of both systems:

> *Ease of using the system*, *perceived usefulness of the system*, *affective attitude towards the system*, and *intention to use the system in the future*.

The evaluation of both systems' user-acceptance was performed based on the Technology Acceptance Model (TAM) (FD Davis, et al 1989). TAM has been successfully used to study user's acceptance of IT systems. In its most basic form, it states that perceived usefulness and perceived ease of use, would determine the behavioral intention to use a system. It also assumes that this behavioral intention could predict the actual use of a system.

Perceived Ease of Use: Defined as the degree to which an individual believes that learning to use a technology will require little effort.

Perceived Usefulness: Examines individual believes that use of the technology will improve performance.

Attitude: Positive feeling or emotion about using the technology.

Intention to use: The likelihood that an individual will use the technology in the future. The main instrument to measure these influences is through the use of questionnaires. Items that measure the same influence can be grouped as a measure of a more general construct. The validation of a model typically includes a long term observation of the actual use of the technology, which makes it possible to relate scores on 'intention to use' to 'actual usage'. However, in this paper the focus is on evaluating the two systems based on the basic constructs of the TAM.

Results

It is evident from the results (fig 6) that for each of the TAM constructs, the mean scores between 'system A' (Expert system) and 'system B' (Recursive system) differed significantly. In particular, perceived usefulness of 'system A' is significantly higher than 'system B'. However, perceived ease of use, is in favor of 'system B' (Recursive system). The user's attitude and Intention to use are significantly higher for 'system A' (Expert system) than for 'system B' (Recursive system).

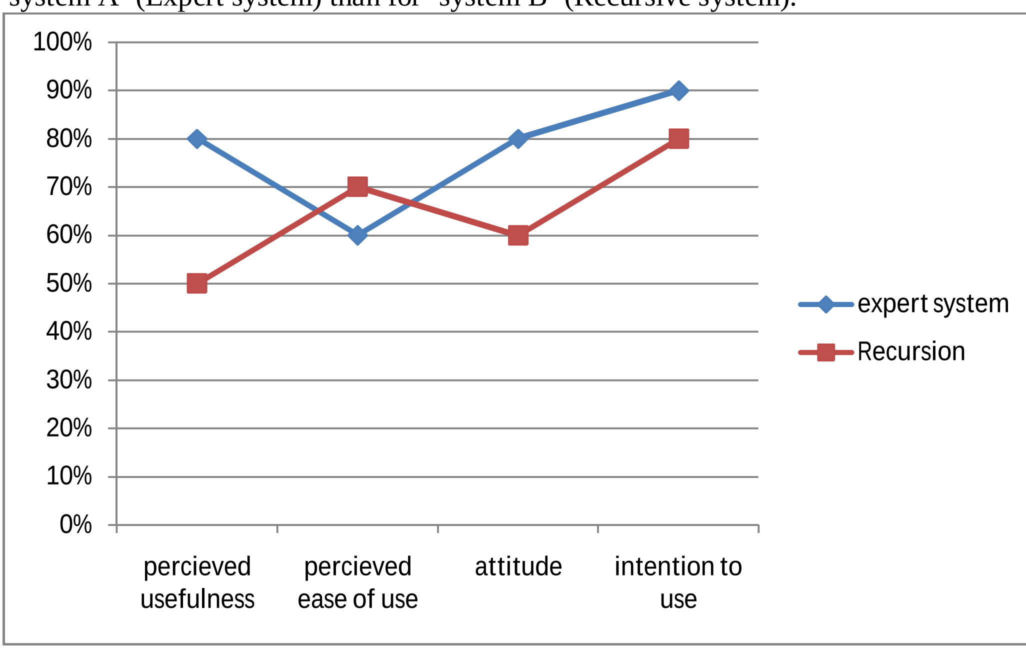

Fig 6 Comparative analysis of Recursion and Expert system implementation of Hanoi

Discussion

System usability owes its roots to the field of Human computer Interaction (HCI). HCI is a discipline concerned with the design, implementation, and evaluation of interactive computer systems for human use. Because systems are built for people and not people for systems, it becomes imperative to build and test systems with a view on the usability of the system.

The result of our study unveils the efficacy of expert systems in providing solutions to practical hands-on problems. Though the tower of Hanoi could be considered as a game/puzzle, it provides a sample scenario of a practical problem with more than one approach towards a solution.

On the other hand, Recursive algorithms have its implementation at the heart of computer science. Though it is efficient for solving a variety of problems, it has its roots in complex search algorithms which have no direct bearing on the physical concerns of humans.

This study unveils significantly the benefit of expert system development for both industry and education. For education, expert systems could be leveraged as a teaching and learning aid. While for industries, expert systems could be leveraged for production, maintenance and trouble shooting of industrial equipments.

Conclusion

The tower of Hanoi which is frequently used in psychological research for solving problems also possess a variant of useful applications from neuropsychological diagnosis and treatment of executive functions, backup rotation scheme for computer backups, to teaching of recursive algorithms for beginning programming students. A rule based expert system approach towards the resolution of the tower of Hanoi presents a non-recursive, but rather interactive and intuitive phenomenon. Results from a comparative analysis show significant usability difference in overall usability of Expert system approach over the traditional Recursive system approach within the confines practical problems such as: resolving the problem of the tower of Hanoi.

References

Dr. Kenneth A Lambert, Dr. Thomas Naps (2001) *Fundamentals of C++ and data Structures Advanced Course.* Second Edition.

FD Davis, RP Bagozzi, PR Warshaw. (1989) User acceptance of computer technology: a comparison of two theoretical models. Management Science, - ebiz.bm.nsysu.edu.tw

Herbert Mayer, Don Perkins (1984) "Towers of Hanoi Revisited".

Hofstadter, Douglas R.(1985) *Metamagical Themas : Questing for the Essence of Mind and Pattern.* New York: Basic Books.

Ivan Moscovich (2001) *1000 playthinks: puzzles, paradoxes, illusions & games*, Workman Pub.

Korf, Richard E., Ariel Felner (2007) *Recent progress in Heuristic Search: A case study of the Four-Peg Towers of Hanoi Problem.*

Michael Negnevitsky (2005) *Artificial intelligence: a guide to intelligent systems.* Second Edition

Newell, A. & Simon, H.A.(1972) *Human Problem Solving.* Prentice Hall, Englewood Cliffs, NJ.

Petković, Miodrag (2009) *Famous Puzzles of Great Mathematicians.* AMS Bookstore. pp. 197.

Sniedovich, Moshe (2010) *Dynamic Programing: Foundations and Principles.*

T. D. Gedeon (1996) *The Cyclic Towers of Hanoi: An Iterative Solution Produced by Transformation. The Computer Journal* 39.